Open Source Integrated Library System in a Rural American Library

By John Brice and Cindy Murdock Ames

LYRASIS

Open Source
Integrated Library System
in a Rural American Library

by John Brice and Cindy Murdock Ames

LYRASIS

First edition: September 2014

This case study was funded by a grant from the Andrew W. Mellon Foundation and is published under the terms of the LYRASIS/Mellon Intellectual Property agreement.

Cover graphic created by Tagul.com.

PDF version generated by LibreOffice 4.3.2.2
ePub version generated by writer2epub 1.1.28 and Calibre 2.3.0

This document is also published on the FOSS4Lib.org website.

Table of Contents

Introduction

Many times the argument is made that a small sized library does not have the resources to do Open Source Software (OSS). This case study was written by a small library that made it happen because it believed that the future of library technology is OSS. Libraries should naturally embrace the OSS philosophy because it is the same philosophy that libraries have always followed: Working collaboratively to create, store, manage and share information.

Back in the 1960s an IT person working for a Fortune 500 company once said that no one ever got fired for selecting IBM as their IT infrastructure, meaning that if things went wrong, the IT person could not be held responsible. Today in the library world, the same philosophy is at work with libraries selecting proprietary integrated library systems (ILS) because it is safe, and if something goes wrong, it's the vendor's fault. Unfortunately, this philosophy has made it difficult to gain large amounts of ground for OSS ILS installations in America.

The following narrative describes the Meadville Public Library's now 13-year odyssey of discovering, selecting and using the OSS Koha ILS. Through our adventures in using OSS we have always followed the philosophy that if something doesn't work right to start with, we can fix it ourselves. It is fortunate that we have that philosophy, as we made numerous miscalculations and guesses that were just plain wrong. Our story can be used to help other libraries discover where the pitfalls lie, hopefully making future libraries' adoption of OSS easier. The following case study is written with the intent to illuminate, educate and inspire others to consider installing OSS in their library.

Background

The Crawford County Federated Library System (CCFLS) is a federated library system of nine state-aided libraries and two reading centers serving 93,000 patrons in Northwest Pennsylvania 40 miles due south of Lake Erie. Please note that the most important word in our name is "federated". In Crawford County, "federated" means that all libraries have to agree. The Meadville Public Library is CCFLS's headquarters and is the largest library in the system: a medium sized public library serving 38,000 patrons in central Crawford County.

Around 2000, the nine member libraries decided it was time to look into acquiring a new ILS. In 1992, CCFLS had originally installed a standalone ILS (Winnebago Circ/Cat) in each of nine member libraries. Though fast, reliable and bulletproof, by the end of the 1990s it became obvious that a text-based ILS or online public access catalog (OPAC) could not compete in a world of graphical and web-based programs. The old ILS architecture was basically a PC in each library running the circulation (circ) station software, along with one OPAC station networked to it. Only the two largest libraries had standalone circulation servers: Meadville and Titusville. Needless to say, not having a centralized catalog made integrating operations across the system very difficult. In addition, Winnebago could not be searched via the web and was difficult to use for the patrons that were used to a graphical point-and-click environment. Also, since it was DOS-based, Winnebago required Windows on staff circulation stations and OPACs.

By 2000, CCFLS had decided to pursue the idea of installing a centralized ILS. The next part of the story was what ILS would we select? The Information Technology Department thought it was important to look only at OSS ILS options while many of the member libraries thought that only a proprietary closed solution was appropriate.

The Case for OSS at CCFLS

The IT department and library administration felt very strongly that it was important to use an OSS ILS. The traditional philosophy librarians have followed concerning the acquisition of ILS software was to purchase off-the-shelf software and fit it into our operations the best way we could. From the perspective of the 1980s, using a circulation system from a third-party closed-source vendor was a smart solution. Programming was difficult, computer hardware was very expensive, support for hardware in a rural city was next to impossible, and there really was only one very large computer to support and manage.

From today's perspective, using third-party closed-source vendors to provide libraries with ILS software is no longer the only solution. Software can be easily written in modern computer languages, and it can be run on inexpensive hardware that is easily supported by individuals who can be trained at local colleges, universities or tech schools. This means that libraries can have individuals on staff that can write or alter OSS to fit their unique needs.

Because of the flexibility of OSS platforms – tools such as LAMP (aka Linux/Apache/MySQL/Perl and/or PHP) form the building blocks for multiple useful services for libraries, like ILS, websites, public computers, web filtering and more – CCFLS has chosen to prioritize hiring capable staff that can integrate OSS tools into library use rather than expending funds on often expensive commercial solutions. Additionally, funds not spent on commercial software and licensing have instead been expended to purchase better hardware and hire staff capable of supporting OSS solutions.

We have three basic reasons why the CCFLS continues to invest in and use OSS:

1. We didn't want others to dictate when we must purchase new equipment. OSS hardware requirements are often much lower than those of proprietary software, meaning that hardware we purchase can have a much longer useful life and can often be repurposed for other projects. We also build our own hardware which is much more economical and usually allows us to buy more powerful hardware than we otherwise could afford. Finally, since a vendor isn't dictating to us when our hardware must be upgraded, we are in control of the upgrade cycle, which makes planning and budgeting more accurate.

2. We were tired of having to work around software. Proprietary ILS often requires altering policies to fit the program's functionality. We wanted to be able to change the ILS to fit our policies. We also wanted to be able to develop new features and have full access to our data, something which was far less likely to be possible with a proprietary ILS. Additionally, with OSS you have access to the full version from the start, allowing you to test it with your own data without having to make a purchase.

3. Using OSS gives us the option of not having to pay for software, licensing or support. Though the right to run OSS is cost-free, it is not necessarily cheap. Library staff still have to spend time and energy learning, configuring, documenting and testing the applications. We have found that over the total life of a project, OSS is considerably cheaper than proprietary solutions. OSS solutions help make our IT budget go further.

CCFLS realizes using OSS is considered risky in that the individuals who develop OSS solutions may leave their institution and take their knowledge with them, thus orphaning the project. Jonathan Rochkind's excellent article "A Primer in Risk" (*Library Journal* November 15, 2008) describes this issue in great detail. Since we have been doing OSS over many years CCFLS has developed an institutional knowledge that allows staff to leave while allowing the project to continue.

Overall, we have found that OSS works well, is reliable and is, when calculated over the total life of a project, a low-cost solution.

Investigating OSS ILS Options

At the time of the ILS selection (2004 to 2005), some of our member libraries asked very tough questions concerning the risk of installing an OSS ILS. To answer their concerns, the IT Department did a great deal of investigation into the two available OSS ILS options: OpenBook and Koha. In the beginning we preferred OpenBook over Koha. Unfortunately, the code for OpenBook was never released so we were not able to test it. This left Koha as the only OSS solution available to us. The Evergreen project, the other main OSS ILS option today, had not even been begun development.

Originally, we had some serious reservations concerning Koha. In order to understand our apprehensions, we need to describe how Koha came into being. The Horowhenua Library Trust (HLT) – a small rural library system in New Zealand– discovered in 1999 that its old circulation system was not Y2K compatible and that commercial software was way too expensive for its budget. So Rosalie Blake, Head of Libraries, approached a local firm called Katipo and proposed they create an OSS ILS. After a lot of discussion and some trepidation, HLT agreed to commit to the project. This project became Koha. The word Koha means "the gift that keeps on giving" in Maori, the New Zealand indigenous language. Within the meaning of the word "koha" (pronounced ko-HA) is also an unspoken understanding of responsibility that this gift is to be shared and enjoyed many times over.[1]

The HLT's original requirements for Koha were rather basic: that it run at a reasonable speed, be easy for staff and patrons to use, have a graphic interface accessible through a web browser, and require little upgrading of hardware, and provide decent financial reporting. Sticking to the idea of OSS, Koha was written using Perl for the LAMP platform.

We first tested an early version of Koha in 2001 and found that it lacked MARC support. It also did not have an acquisitions module, had no reporting module, and the fines system was very different from how we were used to calculating and tracking fines. In 2002, Nelsonville Public Library in Athens, Ohio, scaled a major hurdle for US Koha adoption by funding a project that added MARC support to Koha. After Nelsonville migrated to Koha, we visited them to see how Koha worked in August 2004.

The trip was informative but helped us identify another big issue with Koha: slow search transaction speeds. We noticed that the response time at Nelsonville from the patron/staff clients to the servers was barely acceptable. Koha had what is commonly referred to as a scalability problem. What worked fine for a library with less than 50,000 items would not work well for a library system with 400,000 items. The main cause of this issue was that all search queries were made directly to the MySQL database. This included every word in every MARC record, patron data and all transactional data. Every search, every checkout had to be resolved through the MySQL tables. The big resource hog was the MARC Index. With databases of less than 400,000 MARC records this was not an issue, however over that limit Koha's speed became extremely slow. Nelsonville addressed the issue by running a separate search server and cached, in memory, popular search terms. CCFLS had a larger MARC database and we did not want to use a work around so we decided the best solution was to modify Koha.

Since we did not want to install software that wouldn't fully meet our needs, and at the time did not have the staff needed to modify the code, the only available option was to hire a firm to modify Koha to eliminate the bottleneck. Eventually we found a firm to do the work for $35,000. That firm was LibLime, founded by Joshua Ferraro, a former employee of Nelsonville Public Library who had been a member of the team that implemented Koha there. Mr. Ferraro suggested we fund the integration of IndexData's Zebra indexing engine into Koha, which would be used to index and search MARC records in Koha, offloading the majority of queries from the MySQL database, relieving the bottleneck, and opening the way for larger libraries to adopt Koha. The IndexData Zebra Index had already been identified by the Koha community as the best way to fix the scalability bottleneck. Zebra was chosen because it too was an OSS product, could be used to offload catalog queries from the MySQL database,

1 http://koha-community.org/about/history

and could support MARC and a variety of other data formats. Zebra was also picked due to its successful use in other large database projects requiring a fast index.

The discovery of the speed issue with Koha caused some concern from the member libraries in the CCFLS system. There had been a great deal of disagreement between libraries when we originally selected Winnebago in 1991. More than ten years later, some of the bad feelings remained between the two camps. It manifested this time in the debate as to whether we should select an OSS ILS or a proprietary ILS. Once the libraries discovered that the "free" OSS option would now cost $50,000 ($35,000 for software modifications and $15,000 for new servers and hardware), the IT Department had to highlight other benefits to convince our libraries to buy into an OSS ILS.

Review Process of ILS Selection

One of our principles in Crawford County is that every member library had to agree which means trying to gain consensus of over 100 people (nine library Boards times ten members each plus the librarians). The skepticism about using OSS meant that we had to consider both proprietary and OSS options. After some negotiations, a process was developed by the CCFLS Board to select a new ILS, with the two contenders being Koha and Sagebrush. The first part of the selection process was demonstrations.

Sagebrush demonstrated first, and it was fantastic. The Sagebrush ILS had a staff interface that was colorful, useful and very easy on the eyes. All of the librarians loved it. Seeing the results from the Sagebrush meeting, the OSS camp realized that the Koha staff interface was clunky and difficult to use in comparison.

Realizing we had to do a fast revision of Koha, our newly hired full-time software developer, Kyle Hall completely redesigned the Koha staff interface to streamline it and make the work processes easier to understand. At the same time, Cindy Ames, our IT Director, loaded our data into a demo installation of Koha so we could show the librarians how it looked in Koha. It also demonstrated that we could easily migrate from Winnebago to Koha and were capable of altering and managing the software in-house.

Admittedly the Koha demo was not as slick as the Sagebrush demo, but the member libraries were impressed to see actual local data in the system and the fact that we could change the program to fit our needs. We also pitched the idea of allowing the libraries to change Koha in any way that the member libraries saw fit.

The CCFLS librarians embraced the idea that they could customize the software to their preferences. The IT Department made a number of promises to modify Koha. Some of these promises would cause us issues in the future as CCFLS began working as a more unified system rather than as individual libraries. However, considering the environment of the time in 2005, these promises were critical for securing the selection of Koha as our ILS.

So after years of discussion, field trips and demonstrations, it came time to make the decision at a joint Board/librarian's meeting in October 2005. Most of the libraries were willing to go with Koha, though there was some concern that we were being too ambitious. In the end, all of the libraries agreed, though one library had serious reservations. So, all the IT Department had to do was follow through on the promises that were made.

Koha Development for CCFLS

To fix the scalability problem caused by the MySQL bottleneck, in 2006 CCFLS wrote an LSTA Grant and subsequently hired a third-party company, LibLime, to integrate Index Data's Zebra indexing engine into Koha. While Koha's indexing engine was being rewritten, CCFLS designed a new staff interface for Koha. After a series of weekly meetings between the implementation staff and CCFLS librarians, we rewrote templates from Nelsonville Public Library to create our own look and feel to Koha. Our on-staff developer also wrote new features for Koha to make it better suit our needs, such as a new system for calculating fines and a custom reports module.

MPL is accustomed to contractors taking longer than projected to complete projects. We had received regular updates and progress statements from LibLime, and we were aware that the vendor was some months behind on the project. However, the last progress milestone was the delivery of usable software, which was tied to the last payment. LibLime informed us that the Zebra indexing was ready, but the search functions within Koha needed refinement. LibLime suggested that we join them and other Koha developers from around the world in Marseille, France, in May 2006 for a week-long hackfest to finish the work. Since we wanted to begin the migration process, it was decided that the CCFLS management and IT Department would go to the hackfest. With less than a month's notice (and two MPL staff members needing to quickly acquire passports), we were on our way to the first Koha Hackfest.

As far as problems are concerned, spending a week in France is one that most of us would like to have. However, the Koha Hackfest was all business. Over a period of five days, 10 software developers spent nine hours a day debugging, rewriting, teaching, planning and documenting what became known as the dev-week version of Koha. Additionally, our developer received an intensive education in Koha's codebase. We also enjoyed many fine meals – the Chinese/Vietnamese restaurant with mint egg rolls is still memorable almost a decade later.

The dev-week Koha version program integrated Index Data's Zebra indexing utility eliminating Koha's MySQL bottleneck. This modification significantly increased the speed of Koha for large libraries and library systems, and it allowed libraries with collections of more than 4 million plus items to use Koha. This new version, officially dubbed Koha Zoom, was first implemented in a public library in Western Provence, France in January 2007. Even though we paid for the Zebra integration, we were not the first library to use the upgraded Koha. However, implementing it in France allowed Koha's developers to fully test and debug it before making it available to us, which we found to be an acceptable tradeoff.

Using dev-week Koha, we then loaded our data onto an old server and began running tests. Using current circulation data, we ran a number of dry runs to test the migration process and to have a fully operating ILS available for staff to train and test. During this time, Cindy developed the parameters for data migration and Kyle wrote the migration scripts. For the data that we couldn't export in csv or MARC format, Kyle wrote scripts to extract the data from reports we ran in Winnebago. These scripts are still available at http://kylehall.info/index.php/projects/koha/koha-tools-download/. Cindy used tools such as xml2marc, MARC::Lint, yaz-marcdump and MarcEdit to clean up the MARC records exported from each CCFLS library's Winnebago database.

At the same time, a team of CCFLS librarians and staff examined the migrated data for errors, tested the operation of Koha, and made decisions on changes to the code that needed to be made. We used the Mantis bug tracker to track issues and development. All software changes were managed locally in git and then merged with the code base from Koha dev-week. One issue discovered during this process was that Winnebago had a basic non-MARC editor for adding and modifying materials, while Koha had a full MARC editor with the possibility of using multiple frameworks. Although Winnebago's basic editor was great back in the 1990s when our rural librarians didn't need knowledge of MARC in order to catalog, we discovered that there was little adherence to cataloging

rules outside of the two larger libraries in the system. This discovery resulted in CCFLS having to do database cleanup on nine separate MARC databases plus training staff on proper MARC cataloging.

At this part of the process, we found that the amount of clean up work plus additional training of staff was going to be huge and more then what our small IT Department staff could handle all at once. One of the benefits of OSS and hosting your own solution is that you can dictate your schedule to match the staffing resources you have. By spreading the implementation schedule over a year, we took what would have been a huge job done all at once into manageable chunks. We decided to migrate Meadville Public Library first, working out any major bugs that came up before beginning to migrate the rest of the libraries one at a time.

Migration to Koha

We began using Koha at Meadville Public Library in May 2007. It was decided that MPL would be first to migrate since IT staff offices were there, making it easier to be on hand for dealing with any unanticipated issues that arose. We shut the old circulation system down at closing on a Friday evening at 6:00 PM, and had Koha up and running with all our data migrated by 7:30 that evening. We closed on Saturday in order to convert all the staff computers and OPACs from Windows to Linux-based thin clients. Since Koha's staff interface and catalog are web-based, it allowed us to use any platform that could run a web browser for both circulation and the public catalog computers. We were already using Linux-based thin clients for our public computers, and were eager to take advantage of the opportunity to convert our circulation and OPAC computers to thin clients as well. When we reopened on Monday, the staff logged into the new Koha ILS and all of the MARC records, patron records and transaction data were available. There was no need to run dual ILS or close out the old Winnebago system.

The other eight CCFLS libraries were migrated to Koha by the end of 2008, averaging about one library per month. Because by choosing Koha we were able to set our own migration schedule, rather than having to do all libraries at once as we probably would have had to do with a vendor, we were able to take our time in migrating all our libraries. Each library had its own database of patron, MARC and circulation data that needed to be integrated into a single database in Koha, while other libraries that had already been migrated were already using that database, so caution was in order.

One of the reasons for the long timeframe was irregularities in the MARC records and other data in most of the member libraries. There were a great many encoding issues in the libraries' MARC records originating from our original retrospective conversion to Winnebago; the encoding issues had to be discovered and fixed one by one, a slow process involving a lot of data processing with xml2marc, MarcEdit, and other MARC editing tools.

Winnebago's database was also structured to only have one item per MARC record, while Koha was structured to have multiple duplicate items attached to a single MARC record, so Kyle had to write a script to merge duplicate items into the same record. Because we had nine separate libraries with different circulation rules merging into a single ILS, we decided that each library's items should be separate. At the time, circulation rules were determined by library and item type, not at the item level, so the merging script had to be run for each library before their data could be imported into Koha.

Some libraries also had used barcode numbers outside of their assigned ranges, making it necessary to re-barcode some items. Additionally, there were quite a few patrons that used more than one of our libraries, so we had to merge a lot of patron records as well. We also asked the libraries to do a cleanup on their databases, deleting extremely expired patron accounts and long lost items before they were migrated, to help eliminate as much old and potentially bad data as possible.

Since we were migrating our nine independent libraries one at a time over the course of more than a year, we had the luxury of training each library's staff just before they came online. Audrey Porter, staff trainer at Meadville, developed the majority of the training materials used, as well as performed most of the staff training. Cindy trained the librarians on cataloging. This slow migration schedule allowed us to focus on training just a few librarians at a time, rather than everyone at once, giving our library staff a chance to work one-on-one with our trainers.

The staff accepted Koha well, though there were some issues especially when we first migrated. Right away the circ desk staff identified some bugs, and everyone had opinions on how the program could be improved. In most cases, we could fix the problem ourselves, usually within a day or two, or make some customizations to the code that added to the capability of Koha. An example of a bug was the discovery that word "not" could not be included in searches in the OPAC, because it was always treated as a boolean operative. In two days, we identified the problem, found the code involved, and fixed the bug. Once fixed and tested, we submitted a patch to the Koha

community, solving the problem for every other library using Koha. In another example, some staff did not like the results page for title searches, so we added code to sort search results in the same manner as our old circulation system.

Management of Code Changes

We knew going into Koha that we would be altering the program to meet our own unique needs, which is one of the major benefits of using OSS. The system we set up was similar to how the Koha community managed its code base and releases. We installed a version of Koha on a test server. When we made changes to Koha, we would commit it locally to git, a source code revision management utility which would track our changes to the codebase, essentially making our own "fork" of Koha. Since we were altering the program a great deal, we would test our changes to Koha through a three tier process: proof of concept, beta testing, and production. Proof of concept was to test how it worked and to see if any further changes were needed. Beta was to make sure it worked as advertised in a production setting and didn't break anything else, and finally production was releasing the code for the front line staff to use.

With a developer on staff, making changes, though not inconsequential, was something that could be done on a regular routine. Once the staff saw that changes could be made rather quickly, the IT department became deluged with requests for changes and additions. Wishing to create goodwill towards Koha, the IT staff worked hard to incorporate the staff's suggestions. One or two changes are manageable, but when you create multiple changes in software, one right after the other, it can create chaos. Additionally, some changes may have unexpected consequences. Some staff members would also make direct requests for changes to our developer rather than bringing the change request up at meetings, adding to the potential for confusion. After a couple weeks of rampant chaos, we finally regained order by implementing a formal change request process using the Mantis bug tracker.

For the first few years not many of our changes were incorporated into the mainline Koha program. The reason was that when we did submit changes we could not get other libraries or companies to sign off on our revisions. It would take a number of years for our developer to gain enough experience and horse trading (you sign off on my changes and I will sign off on yours) in order to get changes into Koha. Not getting the changes incorporated into mainline Koha would cause us additional problems down the road.

Koha's development community is extremely active, so after we initially migrated to Koha in 2007/2008, we wanted to install the latest improved version of Koha. Unfortunately, we found that we had altered Koha so much that we had made it difficult to upgrade. We realized that our strategy had a serious flaw in that it made upgrading to new versions of Koha difficult because we had to reintegrate all of our custom code, and then extensively test the changes, all of which took up to a year. After going through a very difficult upgrade path in 2010, we made the decision to abandon our custom templates in favor of using the default Koha templates with a few minor modifications.

Just switching over our templates was not enough. We still wanted to use our custom changes, but eliminate the need for rewriting the code with every update. So we developed a few strategies. First, for major changes in Koha, we submitted them to the community. We have made a great deal of progress in incorporating our changes into Koha's main codebase. These include: the MARC Modification templates, Rotating Collections, Clubs and Services, and the Plugins system. All of our submissions have been incorporated so far except the Club and Services module, which is now waiting for approval. Secondly, for changes that are deeply integrated into the main codebase, our programmer created the Plugins system. This allows us and all other Koha libraries to create custom modules for Koha that can be easily installed on a standard Koha installation without disrupting or altering its functionality. We have created many plugins, such as custom reports, spine label printers, and a tool for generating MARC records and barcodes for a year's worth of magazine subscriptions at a time. Additionally, wherever possible changes are designed to be easily enabled or disabled with a system preference, which helps with getting our major changes integrated into mainline Koha, since libraries that don't want a new feature can simply disable it.

One major change, over the past two years, is that our programmer has been working as a full-time Koha developer for Bywater Solutions while still working part-time on Koha and other projects for CCFLS. While we did

lose some of his time with the change, we did gain better access for incorporating CCFLS's changes into Koha. When Kyle worked for us exclusively, it was very difficult to have other developers review, test and incorporate the changes into Koha. With him working as a developer for a major vendor in the community, getting our changes integrated into Koha has become easier. With all of our major custom changes now in or scheduled to be in the Koha ILS, we are planning to upgrade to the current version of Koha in September 2014. This will also allow us to upgrade Koha on a regular annual schedule so we will be able to roll out the latest ILS features as soon as they become available.

Costs

Our costs in migrating to Koha are typical compared to other OSS projects we have implemented. The development costs for OSS are similar to the purchase of proprietary solutions. However, the ongoing expense for OSS is only for hardware updates, so costs are easily predictable and lower than commercial solutions. In addition, OSS costs show a large savings from the fourth through the seventh years of operation, as there is no large "upgrade" or new version costs. The real savings in OSS is not the acquisition cost, but the yearly operating savings, which can be significant over a ten-10-year period.

The costs for Koha came within our expectations. We did spend as much for Koha as we would have implementing a proprietary product. The cost of the upgrade to the Koha software was more than $35,000. We also spent slightly more than $15,000 on new servers and other hardware equipment. So the overall direct "acquisition" cost was slightly above $50,000. Since we support the system ourselves, we have no ongoing licensing fees or support fees. Every few years we budget for two servers, one of which serves as a live backup as well as a host for running MySQL-intensive reports. The upgrades cost us between $7,500 and $10,000 every three years total. Overall, over the last eight years, our total direct costs have been $77,000 or roughly $7,700 a year. Considering we have more than nine libraries using our Koha ILS, this is less than $1,000 a year per library. Of course if you add indirect costs, the total becomes somewhat higher; however, our IT Department staff is the same size now as it was when we started this project (2.5 FTEs). It is easy to conclude we would have spent the same amount of staff time managing a closed source ILS, making indirect costs a wash. Additionally, the funds spent toward integrating the Zebra index into Koha paved the way for other libraries our size and larger to be able to adopt Koha, leveraging our investment in a way that cannot be done using closed source software. Libraries our size adopting Koha today would not have the expenses we did in migrating, unless they were to sponsor major development as we did.

Instead of expending funds on licensing or commercial support, CCFLS has prioritized investing in staff that is capable of implementing, developing, and supporting multiple OSS projects (not just Koha, but other programs like Libki), so that we get more "bang for our buck" from our IT staff.

The Importance of the Koha Community

A major criticism of OSS is that there are no guarantees that a particular project will survive into the future. The key to a successful OSS project is an active community of users and developers. Koha has an extremely active worldwide base of both users and developers, and a rapid development cycle. In our experiences, overall, we have found OSS to be more reliable, and we frequently receive faster support than with proprietary software. Usually with a successful OSS project, there is a large community from which to draw expertise and you can often communicate directly with the developers themselves.

If you wish to support Koha internally, you ideally need access to someone, either staff members or consultants, who understand Linux and are comfortable using a command line. Fortunately, these skills can be learned. Our IT Director has an undergraduate degree in Art History and a MLIS. All of her Linux skills have been learned on the job. We also have access to a programmer to help support and modify our OSS programs. Kyle has a Master's Degree in Information Technology, which he earned locally at Edinboro University of Pennsylvania.

Another option is to hire a commercial service to help install, manage and support your OSS ILS. CCFLS has in the past purchased support from a commercial Koha service. We hired them because we thought we might need help as we migrated, but we found we could handle the vast majority of the tasks ourselves. We did rely on them for a couple of issues and to help us figure out how to fix them. However, we ended up correcting many issues ourselves as we learned more about Koha. Since at that time the vendor was still actively working on the version of Koha we were using, we looked on it as an additional means of supporting development.

Conclusion and Lessons Learned

Any ILS project is complicated, so you may not want your first OSS project to be an ILS. Consider gaining experience with implementing other OSS solutions in your library first.

Installing OSS can be difficult, and there will be, in all likelihood, politics involved in the decision, especially with software as mission-critical to a library as an ILS. The best way to deal with the politics is to be truthful and not oversell the promises. However, having confidence in your team and the idea that any problem can and will be solved is a great sales pitch. Of course you have to back it up (see above).

Installing a new ILS and adding features via software development is even more complicated than an ILS project.

Frontline staff must realize that there will be problems and that they need a communication channel with management to express their concerns about problems with the software. A process should be in place for reporting problems, such as a bug tracking program like Mantis or Bugzilla, and everyone should be aware of the proper procedures to follow.

Upper management has to support an OSS ILS and understand that problems will occur that do not necessarily occur with a closed source solution. CCFLS believes that advantages of OSS far outweigh the negatives. The process of getting support for OSS can be very different than with proprietary software, particularly if you are supporting the software yourself rather than paying for commercial support. With a proprietary ILS, you typically contact technical support for assistance. There is also generally only one channel for getting support-- the company producing the software. If you are unhappy with the support you're receiving for that product, you don't have many alternatives. With OSS, there are a variety of ways to get support. Particularly with Koha, there is online documentation, mailing lists, IRC channels (where the developers commonly hang out), etc., in addition to commercial support from a number of vendors. If you're unhappy with the support you're receiving from one company, you are free to choose another. You also have the ability to view and/or modify the source code so that as long as you are capable of doing so, you can directly dig into your ILS to see what is going on.

Changing the code in an OSS ILS is not terribly difficult; just realize that if you change the program once, you will have to reintegrate your code every time you upgrade. So, upgrades will take longer and require a developer. Of course if you can get your changes incorporated into the main code base of the program or write them in such a way as to isolate them from the main code base (e.g. a plugin), you do not have to reintegrate your custom code every time there is an upgrade.

You need to budget time and money to support the OSS ILS community. When selecting an OSS ILS, you are not just picking software, but also becoming a member of a community. When considering an OSS ILS, you need to evaluate the community and understand how it works, meets, makes decisions and handles updates. As a member of a community, an OSS library has a responsibility to participate by volunteering and potentially contributing funds for documentation, future features and enhancements. Of course, you should consider sharing code you have developed internally and meeting with participating libraries and software developers. Though you do not have to pay for support directly, there are a number of indirect costs that need to be considered when selecting an OSS ILS, as noted above.

Appendix: Technical Specifications

The hardware requirements for a Koha server are directly related to the collection size of the libraries using it, as well as the amount of circulation traffic generated by those libraries. CCFLS currently has 35,688 patron records and 306,496 bibliographic records for 326,475 items, with about 440,000 circulation transactions per year. Our current server specs are:

- 16 gigabytes memory
- 2 dual-core AMD Opteron 2214 processors
- 6 500 gigabyte hard drives, configured like so:
    - 2 in a RAID1 array for the Zebra index: since the Zebra index is constantly being written to, we found that giving it its own separate RAID array improved performance significantly. We chose to use RAID for extra reliability, since even though the data stored in the Zebra index is not unique, if the drive storing it goes down, it requires a rebuild of the index that can take many hours, during which catalog searches are not possible.
    - 4 drives with two partitions each, arranged in two RAID1 arrays: 3 are active and one is a live spare. The RAID arrays are:
        - md0 mounted as the /boot directory, using 11% of 228 MB
        - md1 mounted as the / (root) directory, using 70% of 459 GB

We have two servers with those specs: one of which is the main Koha server, and the other is a copy of it utilizing MySQL replication, so that we can use it as a hot spare if the main server goes down. The secondary server is also used for running reports, which can be very demanding on hardware. We also initially configured our server to use https encryption for the staff interface. Since we do not have a WAN, our circ transactions are travelling over the Internet. We found that this slowed down transaction speed significantly, so we developed a hardware-based encryption platform to offload the encryption. For our next server upgrade, we are considering switching to a VPN for circ traffic.

The servers we purchased for our upgrade this fall have the following specs:

- 32 gigabytes memory
- 2 AMD Opteron 6212 Interlagos 2.6 ghz 8-core processors
- 2 120 gigabyte SSD drives for the Zebra index, configured in a RAID1 array
- 4 1 Terabyte hard drives configured in RAID10 arrays as above, almost doubling our storage capacity

For backups, we have a backup server that utilizes rsnapshot, a backup utility using rsync. We make backups of the raw MySQL database files, perform backups with MySQLdump (which make for easy database restoration), and do a flat backup of the servers' file systems.